Philosophia Scientiæ
Travaux d'histoire et de philosophie des sciences

16-3 | 2012
Alan Turing

La thèse de l'hyper-calcul : enjeux et problèmes philosophiques

Florent Franchette

Édition électronique
URL : http://journals.openedition.org/philosophiascientiae/766
DOI : 10.4000/philosophiascientiae.766
ISSN : 1775-4283

Éditeur
Éditions Kimé

Édition imprimée
Date de publication : 1 novembre 2012
Pagination : 17-38
ISBN : 978-2-84174-603-3
ISSN : 1281-2463

Référence électronique
Florent Franchette, « La thèse de l'hyper-calcul : enjeux et problèmes philosophiques », *Philosophia Scientiæ* [En ligne], 16-3 | 2012, mis en ligne le 01 novembre 2015, consulté le 30 avril 2019. URL : http://journals.openedition.org/philosophiascientiae/766 ; DOI : 10.4000/philosophiascientiae.766

La thèse de l'hyper-calcul : enjeux et problèmes philosophiques

Florent Franchette

IHPST, Paris 1 Panthéon-Sorbonne (France)

Résumé : Dans cet article je réponds à deux questions philosophiques soulevées par la thèse suivante appelée « thèse de l'hyper-calcul » : il est possible de construire physiquement un modèle d'hyper-calcul. La première question est liée aux enjeux de cette thèse. Puisque la construction physique d'un modèle de calcul dépasse le cadre mathématique initial de la théorie de la calculabilité, j'expliquerai pourquoi il est nécessaire de construire physiquement un modèle d'hyper-calcul. La seconde question concerne le problème de la vérification : à supposer que l'on dispose d'un modèle d'hyper-calcul construit physiquement, il serait impossible de vérifier que ce modèle calcule une fonction non calculable par machine de Turing. Je proposerai une analyse de ce problème dans le but de montrer qu'il ne remet pas en cause de façon explicite la thèse de l'hyper-calcul.

Abstract: In this paper I answer two philosophical questions raised by the following thesis called "hypercomputation thesis": it is possible to physically build a hypercomputation model. The first question is connected to the stakes of this thesis. Since the physical construction of a computational model goes beyond the mathematical framework of the computability theory, I will explain why it is necessary to physically build a hypercomputation model. The second question concerns the verification problem raised against the hypercomputation thesis: if we assume that we have a hypercomputation model physically built, it would be impossible to verify that this model is able to compute a function which is not computable by a Turing machine. I will propose an analyze of this problem in order to show that it does not explicitly dispute the hypercomputation thesis.

Philosophia Scientiæ, 16 (3), 2012, 17–38.

Introduction

Alan Turing est généralement connu en informatique et en logique pour avoir conçu le modèle de calcul aujourd'hui appelé « machine de Turing » [Turing 1936]. La machine de Turing occupe en effet une place centrale dans ces deux domaines. En informatique, la machine de Turing est le modèle théorique des ordinateurs modernes, tandis qu'en logique, elle est la formalisation de la notion de fonction calculable.

On considère généralement qu'une fonction est calculable lorsque l'on dispose d'une méthode qui permet de la calculer. Une telle méthode, appelée en logique « procédure effective », définit précisément une séquence d'opérations, de sorte que chaque opération soit explicitement définie et que l'exécution de la séquence se termine dans un temps fini. Ainsi, pour montrer qu'une fonction est calculable, il suffit d'exhiber une procédure effective qui permet de la calculer. En revanche pour déterminer si une fonction n'est pas calculable, il ne suffit pas de tester l'ensemble des procédures effectives dont nous disposons puisqu'il serait impossible de savoir si une fonction calculable en l'état de nos connaissances ne se révèlerait pas calculable par la suite. Il est par conséquent nécessaire, dans le cas où une fonction n'est pas calculable, de disposer d'un outil formel permettant d'établir qu'il n'existe pas de procédure effective capable de la calculer.

Dans le but de formaliser un prédicat dont la signification informelle est celle de « peut être calculé par une procédure effective », Turing proposa en 1936 le prédicat formel de « calculé par machine de Turing » ou « Turing-calculable » [Turing 1936]. La machine de Turing est un modèle de calcul dont la puissance est résumée dans la « thèse de Church-Turing » : les fonctions calculables par procédure effective sont calculables par machine de Turing. D'après cette thèse qui est largement acceptée par la communauté scientifique, la machine de Turing délimite les fonctions intuitivement calculables, car si une fonction n'est pas Turing-calculable, aucune procédure effective n'est capable de la calculer (contraposée de la thèse). En particulier, Turing a démontré qu'il existait des fonctions non calculables par machine de Turing [Turing 1936, 145].

Turing est pourtant aussi à l'origine d'un autre modèle de calcul appelé « O-machine », modèle qui a la possibilité d'aller « au-delà » de ce qui est calculable par machine de Turing [Turing 1939, 167]. Une O-machine est une machine de Turing équipée d'un « oracle », à savoir une boîte noire dont le fonctionnement interne n'est pas spécifié, qui est capable de fournir pour certains arguments les résultats de fonctions non Turing-calculables. L'O-machine, de par son architecture et sa puissance calculatoire, n'est pas un modèle de calcul au sens de la théorie du calcul effectif, mais est un modèle d'« hyper-calcul ».

L'hyper-calcul peut être défini comme le calcul de fonctions ne pouvant pas être calculées au sens de Turing, à savoir les fonctions non Turing-calculables [Copeland 2002c, 461]. Cette notion renvoie à l'idée que les procédures ef-

fectives ne sont pas les seules méthodes possibles pour calculer des fonctions mathématiques et que certains modèles pourraient calculer davantage de fonctions que la machine de Turing. En ce sens, l'O-machine est un modèle d'hyper-calcul ou « hyper-modèle », car elle est capable à l'aide d'une méthode non effective (l'appel de l'oracle) de calculer des fonctions non Turing-calculables.

Contrairement à la théorie du calcul effectif, l'hyper-calcul ne fait pas l'unanimité parmi les scientifiques et les philosophes. Même si d'un point de vue logique de nombreux hyper-modèles ont été proposés [Copeland 2002c], les problèmes actuels concernent davantage le domaine physique. Ces problèmes concernent en effet la question de savoir si un hyper-modèle pourra être un jour construit physiquement [Kieu 2002], [Shagrir & Pitowski 2003], [Davis 2006], [Hagar & Korolev 2006]. Plus précisément, le débat consiste à déterminer si l'affirmation suivante que je nommerai « thèse de l'hyper-calcul » est vraie ou fausse : il est possible de construire physiquement un hyper-modèle.

À première vue, cette thèse repose uniquement sur la physique, car elle est dépendante de l'existence d'une théorie physique compatible avec la construction d'un hyper-modèle [1]. La thèse de l'hyper-calcul soulève cependant de réelles questions philosophiques. Je souhaite en particulier répondre dans le présent article à deux de ces questions.

La première question est liée aux enjeux de la thèse de l'hyper-calcul. Les enjeux d'une telle construction ne sont pas évidents puisque la construction physique d'un modèle de calcul, qu'il soit équivalent ou non à la machine de Turing, dépasse le cadre initial de la théorie mathématique du calcul effectif. D'une part, la thèse de Church-Turing n'énonce rien en ce qui concerne la puissance de calcul de la machine de Turing si elle est physiquement réalisée. D'autre part, même si l'hyper-calcul est une notion dépassant le cadre du calcul effectif, ce dépassement reste à l'intérieur du domaine mathématique contrairement à la possibilité de l'hyper-calcul dans le domaine physique. Par conséquent pourquoi sortir du cadre mathématique de l'hyper-calcul pour se tourner vers la construction physique d'un hyper-modèle ?

La seconde question concerne un problème épistémologique énoncé contre la construction physique d'un hyper-modèle. Ce problème que je nommerai « problème de la vérification » fut énoncé explicitement par Copeland [Copeland 2002c, 490], puis fut repris par Davis dans sa critique de l'hyper-calcul [Davis 2006]. Le problème de la vérification peut être défini de la manière suivante : supposons que nous diposons d'un hyper-modèle construit physiquement, pouvons-nous vérifier que ce dernier calcule au moins une fonction non Turing-calculable ? La réponse est négative puisque pouvoir vérifier chaque

1. À titre d'exemple, Pitowski [Pitowski 1990], Hogarth [Hogarth 1992] et Earman et Norton [Earman & Norton 1993] ont montré que certains processus permettant de calculer davantage de fonctions que la machine de Turing étaient compatibles avec la théorie de la relativité générale.

résultat calculé par l'hyper-modèle reviendrait à disposer d'une procédure effective nous permettant de suivre le calcul de la donnée initiale au résultat, ce qui est impossible d'après l'hypothèse selon laquelle la fonction n'est pas Turing-calculable.

La conséquence de cette réponse négative est que si nous disposions d'une machine physiquement construite capable d'hyper-calcul, il serait impossible de déterminer si cette machine est réellement un hyper-modèle. En d'autres termes, toute tentative de construction d'un hyper-modèle est vouée à l'échec, car il sera impossible de vérifier que l'on a réussi. Le problème de la vérification est ainsi un sérieux obstacle à la thèse de l'hyper-calcul. Mon but sera toutefois de proposer une analyse plus détaillée de ce problème et de montrer qu'il ne remet pas en cause de façon explicite la thèse de l'hyper-calcul.

Dans la section 1, je montrerai à partir des travaux de Copeland comment définir deux hyper-modèles capables de calculer des fonctions non Turing-calculables. Je présenterai ensuite dans la section 2 la thèse de l'hyper-calcul ainsi que deux tentatives qui ont été proposées afin de la démontrer. Dans la section 3, je traiterai des enjeux de la thèse de l'hyper-calcul en expliquant pourquoi il est nécessaire de construire physiquement un hyper-modèle. Enfin, j'étudierai dans la section 4 le problème de la vérification et je proposerai une analyse de ce problème dans le but de montrer qu'il ne remet pas en cause de façon explicite la thèse de l'hyper-calcul.

1 Calcul effectif et hyper-calcul

Le travail des logiciens des années 1930 tels qu'Alan Turing, Alonzo Church et Stephen Kleene fut de définir formellement l'ensemble des fonctions $f : \mathbb{N}^p \rightarrow \{0, 1\}$ que l'on considère intuitivement comme calculables, où $p \in \mathbb{N}$ est l'arité de la fonction.

On considère généralement qu'une fonction est calculable lorsqu'il existe une méthode, appelée en logique « procédure effective » permettant de calculer le résultat de la fonction pour chacun de ses arguments. La notion de procédure effective est une notion informelle et n'a pas de définition universellement admise. Rogers définit par exemple une procédure effective ou algorithme en utilisant la notion de déterminisme :

> Un algorithme est une procédure mécanique (c'est-à-dire déterministe) qui peut être appliquée à toutes les données en entrée appartenant à une certaine classe et qui conduit éventuellement, pour chacune de ces données en entrée, à une donnée en sortie correspondante. [Rogers 1987, 1]

D'une manière différente, Stone définit un algorithme indépendamment du déterminisme : c'est un ensemble de règles qui définit précisément une

séquence d'opérations, de sorte que chaque règle soit effective et définie et que la séquence se termine dans un temps fini [Stone 1972].

Dans le but de formaliser cette notion intuitive, Turing proposa dans les années 1930 un modèle mathématique de calcul, nommé aujourd'hui « machine de Turing ». Une machine de Turing peut être vue comme une procédure de calcul qui comporte une unique structure de donnée : une suite de symboles inscrite dans les cases d'un ruban potentiellement infini. Les opérations disponibles permettent à la procédure de déplacer une tête de lecture à gauche ou à droite de la suite, d'écrire un symbole à la place du symbole lu ou de ne rien faire. Ces opérations sont très simples et primitives, mais selon Turing elles sont suffisantes pour calculer toutes les fonctions de $\mathbb{N}$ dans $\mathbb{N}$ calculables par procédure effective.

De façon canonique, ce dernier énoncé est remplacé au sein de la littérature par la « thèse de Church-Turing » : les fonctions calculables par procédure effective sont calculables par machine de Turing (Turing-calculables). Cet énoncé est une thèse et non un théorème, car il n'a pas encore été démontré mathématiquement [2]. Toutefois, aucun contre-exemple n'a pu être fourni et la grande majorité de la communauté scientifique pense qu'il n'en existe pas [3].

En acceptant cette équivalence entre les notions de procédure effective et de machine de Turing, Turing a pu prouver que certaines fonctions ne sont pas calculables en démontrant qu'elles ne sont pas Turing-calculables [Turing 1936, 145]. L'exemple paradigmatique d'une fonction non Turing-calculable est la fonction arrêt : soient $\psi_1, \psi_2, \psi_3 \ldots$ une énumération des fonctions calculables par machine de Turing, la fonction arrêt h est définie par $h(x, y) = 0$ si $\psi_x(y)$ diverge pour y et $h(x, y) = 1$ autrement.

À partir de l'existence de fonctions non Turing-calculables, telles que la fonction arrêt, la machine de Turing peut être considérée comme délimitant ce qui est calculable au sens intuitif de ce qui ne l'est pas. Serait-il cependant possible de calculer davantage de fonctions que la machine de Turing ? Ne peut-on pas envisager d'autres types de procédures nous permettant de calculer au-delà des procédures effectives ? Nous pouvons trouver au sein du travail de Turing les prémisses de telles idées.

Turing a introduit dans sa thèse de doctorat une machine différente de la machine de Turing nommée O-machine [Turing 1939, 167]. Une O-machine est

2. La réciproque de la thèse, à savoir l'affirmation « les fonctions calculables par machine de Turing sont calculables par procédure effective » est quant à elle démontrable mathématiquement [Shoenfield 1967, 109].

3. Un argument très fort en faveur de la thèse de Church-Turing réside dans le fait que toutes les formalisations de la notion de procédure effective qui ont été proposées définissent le même ensemble de fonctions. Nous pouvons citer par exemple les fonctions λ-définissables [Church 1936], les fonctions récursives [Kleene 1936], les langages générés par les systèmes canoniques [Post 1941] et plus récemment les fonctions calculées par UMR (*Unlimited Register Machine)* [Shepherdson & Strurgis 1963].

une machine de Turing équipée d'un « oracle », c'est-à-dire d'une boîte noire dont l'architecture n'est pas spécifiée. La particularité de l'O-machine est que son oracle est capable de fournir les résultats de certaines fonctions non Turing-calculables. Plus précisément, si une fonction est Turing-calculable l'oracle n'intervient pas. Mais dans le cas où la fonction n'est pas Turing-calculable, l'O-machine rentre dans un état interne particulier faisant intervenir l'oracle qui fournit le résultat de la fonction pour l'argument désiré. Turing ne donne pas d'informations supplémentaires quant à l'architecture de l'O-machine, mais affirme que le calcul de cette dernière n'est pas effectif :

> Nous n'irons pas plus loin concernant la nature de cet oracle sauf qu'il ne peut pas être une machine [une machine de Turing]. [Turing 1939, 167]

De ce point de vue, Turing n'est pas uniquement le pionnier du calcul effectif, mais aussi de ce que l'on nomme aujourd'hui « hyper-calcul ».

L'hyper-calcul dénote le calcul de fonctions ne pouvant pas être calculées au sens de Turing, à savoir les fonctions non Turing-calculables [Copeland 2002c, 461]. Cette notion renvoie à l'idée que les procédures effectives ne sont pas les seuls moyens possibles pour calculer des fonctions mathématiques et que certains modèles pourraient calculer davantage de fonctions que la machine de Turing. En ce sens, l'O-machine est un modèle d'hyper-calcul ou « hyper-modèle », car elle est capable, à l'aide d'une méthode non effective (l'appel de l'oracle), de calculer certaines fonctions non Turing-calculables.

Plus précisément, l'hyper-calcul peut être défini à partir de la notion de procédure effective [Copeland 2002b]. Pour ce faire, Copeland définit une procédure effective M comme une méthode de calcul devant satisfaire les contraintes suivantes :

1. M doit avoir un nombre fini de symboles et d'instructions.
2. M doit contenir un nombre fini d'étapes.
3. M doit pouvoir être exécutée dans un temps fini.
4. Un être humain doit pouvoir suivre M étape par étape, de la donnée initiale au résultat indépendamment des contraintes de temps et d'espaces mémoire.
5. M doit pouvoir être exécutée par un être humain sans l'aide d'aucune machine physique, indépendamment des contraintes de temps et d'espaces mémoire.
6. Un être humain doit pouvoir exécuter chaque étape de M de manière effective, c'est-à-dire sans ingéniosité ni intelligence.

À partir du fait que ces contraintes sont satisfaites si M désigne une machine de Turing, Copeland montre ensuite que l'on peut formaliser des hyper-modèles s'ils ne satisfont que certaines de ces contraintes :

Machine de Turing accélérante

La « machine de Turing accélérante » (notée MTA) est un hyper-modèle conçu en permettant au calcul de comporter un nombre infini d'étapes tout en restant fini [4] [Copeland 2002a]. La MTA est définie de façon similaire à la machine de Turing, sauf concernant le nombre d'étapes de calcul qu'elle peut exécuter. Tandis que la machine de Turing s'arrête toujours après un nombre fini d'étapes, la MTA a la possibilité de s'arrêter après un nombre infini d'étapes. Cette possibilité repose sur le principe suivant : le temps d'une étape de calcul est deux fois moins important que le temps de l'étape précédente. Plus précisément, si un calcul prend une unité de temps pour exécuter sa première itération, le temps total du calcul peut être exprimé par la série géométrique

$$\sum_{i=0}^{n} \frac{1}{2^i}$$

où i est la présente itération et n est le nombre d'itérations du calcul. Par conséquent, lorsque i tend vers l'infini, le temps total du calcul approche 2, car

$$\sum_{i=0}^{\infty} \frac{1}{2^i} = 2.$$

Si le calcul peut être exécuté au moyen d'un nombre fini d'étapes, la MTA calcule exactement les mêmes fonctions que la machine de Turing. En revanche, si le calcul demande une infinité d'étapes afin d'arriver au résultat, la MTA double sa vitesse à chaque étape ce qui lui permet de surpasser la puissance d'une machine de Turing.

L'avantage crucial de la MTA par rapport à la machine Turing repose sur le fait que la MTA s'arrête toujours. En effet, une machine de Turing qui ne s'arrête pas peut être interprétée de deux façons : soit elle s'arrêtera, car elle va trouver un résultat, soit elle ne s'arrêtera jamais, car il n'existe pas de résultat. Intuitivement, c'est l'impossibilité d'interpréter le non-arrêt de la machine de Turing qui est la cause de la non Turing-calculabilité de certaines fonctions. La MTA quant à elle s'arrêtera toujours puisque qu'elle aura parcouru en un temps fini l'ensemble infini des étapes calculatoires possibles de la machine de Turing.

4. La possibilité d'exécuter un nombre infini d'étapes en un temps fini est un problème philosophique depuis l'Antiquité et a pour origine les paradoxes de Zénon sur l'infini. Ce problème fut l'objet d'un véritable débat au XX^{e} siècle entre Thomson [Thomson 1954] et Benacerraf [Benacerraf 1962] et fut plus récemment traité par Van Bendegem [Van Bendegem 1994].

Machine de Turing à oracle

De la manière dont Turing l'a définie, la machine de Turing à oracle ou O-machine n'est pas fondée explicitement sur la suppression d'une contrainte qui compose la notion de procédure effective. C'est pourquoi Copeland et Proudfoot ont proposé une autre définition de l'O-machine en supprimant la contrainte de finitude du nombre de symboles. De leur point de vue, une O-machine est une machine de Turing composée de deux éléments : un dispositif capable d'exécuter des mesures d'une précision arbitraire et un espace mémoire qui contient une valeur précise appelée τ [Copeland & Proudfoot 1999, 103]. Le nombre τ est un nombre représenté par une suite infinie de 0 et de 1 représentant les valeurs d'une fonction non Turing-calculable [5]. Si cette fonction est notée d, le n-ième symbole de τ représente $d(n)$, à savoir 0 ou 1. Et si nous souhaitons avoir accès au résultat $d(239208)$, le dispositif mesure le 239208^{e} symbole de τ et en fournit la valeur. Par conséquent une O-machine ayant en mémoire un nombre dont les décimales codent les résultats d'une fonction non Turing-calculable peut calculer davantage de fonctions que la machine de Turing.

Même si l'on est actuellement capable de concevoir des hyper-modèles, preuve que l'hyper-calcul est possible dans le domaine logique, l'hyper-calcul ne fait néanmoins pas l'unanimité parmi les scientifiques et les philosophes. Aujourd'hui, le véritable débat autour de l'hyper-calcul ne se situe pas au niveau logique mais au niveau physique, car il concerne la possibilité de construire physiquement un hyper-modèle.

2 La thèse de l'hyper-calcul

Depuis une dizaine d'années, la construction physique d'un hyper-modèle suscite un vif intérêt [Kieu 2002], [Shagrir & Pitowski 2003], [Davis 2006], [Hagar & Korolev 2006]. L'hyper-calcul étant possible dans le domaine logique, la question actuelle que se posent les chercheurs du domaine consiste à déterminer si l'hyper-calcul est possible dans le domaine physique. Plus précisément, les défenseurs de l'hyper-calcul soutiennent la thèse suivante que je nommerai « thèse de l'hyper-calcul » : il est possible de construire physiquement un hyper-modèle.

La possibilité de construire un hyper-modèle nécessite de distinguer deux types d'hyper-modèles :

5. Les résultats d'une fonction de $\mathbb{N}$ dans $\mathbb{N}$ peuvent en effet être représentés par un nombre réel. La première définition que donne Turing du concept de calculable concerne justement les nombres calculables : « D'après ma définition, un nombre est calculable si ses décimales peuvent être inscrites par une machine » [Turing 1936, 116]. Ainsi un nombre est calculable si une machine de Turing peut énumérer une à une ses décimales.

Définition 1 (Hyper-modèles conceptuels et physiques)

- Un hyper-modèle conceptuel est un modèle mathématique qui doit être formalisé sans impliquer de contradiction logique.
- Un hyper-modèle physique est un modèle conçu dans une théorie physique et qui doit être compatible avec les lois énoncées par cette théorie. Si l'on dispose d'une réalisation physique de cet hyper-modèle, nous dirons que l'on a construit l'hyper-modèle physique.

Pour montrer que la thèse de l'hyper-calcul est vraie, ses défenseurs doivent procéder en trois étapes. Tout d'abord, un hyper-modèle conceptuel doit être conçu afin de prouver qu'il est capable en théorie d'aller au-delà de la machine de Turing. À partir de cet hyper-modèle conceptuel, un hyper-modèle physique doit être proposé dans une théorie physique et être compatible avec les lois énoncées par cette théorie. Enfin, l'hyper-modèle physique doit être construit. La première étape ayant été accomplie en partie grâce aux travaux de Turing, il est actuellement nécessaire de déterminer si nous pouvons concevoir un hyper-modèle qui est compatible avec une théorie physique.

Dans ce but, deux stratégies ont vu le jour [6].

Définition 2 (Stratégie I et stratégie E)

- La stratégie I (comme interne) consiste à proposer une théorie physique dans laquelle un hyper-modèle calculera de façon interne, c'est-à-dire sans utiliser d'information externe non calculable issue de la nature (l'univers) [Pitowski 1990], [Kieu 2002], [Shagrir & Pitowski 2003].
- La stratégie E (comme externe) consiste à proposer une théorie physique dans laquelle un hyper-modèle pourra utiliser de l'information externe non calculable issue de la nature. Dans ce cas, l'information issue de l'univers est vue comme un oracle qui permet d'apporter un élément supplémentaire afin de transcender les limites de la machine de Turing [Copeland & Proudfoot 1999], [Calude 2004], [Loff & Costa 2009].

Afin de mieux comprendre ce que signifient les stratégies I et E, apportons des précisions supplémentaires. Pour cela, je présente dans ce qui suit les tentatives de conception de la MTA et de l'O-machine qui illustrent respectivement les stratégies I et E.

Les tentatives de construction physique d'une MTA sont fondées sur la stratégie I, à savoir construire un hyper-modèle physique qui permet de calculer de la même manière qu'une MTA sans utiliser d'information externe non calculable provenant de la nature. La conception d'un modèle physique d'une MTA a commencé à partir des travaux de Pitowski [Pitowski 1990], Hogarth [Hogarth 1992] et Earman et Norton [Earman & Norton 1993]. Ces derniers ont montré que l'exécution d'un nombre infini d'étapes de calcul en un

6. Nous pouvons trouver ces deux stratégies sous la forme de déclinaisons de la thèse de l'hyper-calcul dans [Loff & Costa 2009, 10].

temps fini était possible à l'intérieur d'un espace-temps relativiste particulier appelé espace-temps de Malament-Hogarth. À partir de ces résultats, Shagrir et Pitowski ont ensuite décrit un hyper-modèle physique pouvant calculer la fonction arrêt des machines de Turing [7] [Shagrir & Pitowski 2003, 88].

Cet hyper-modèle appelé HM par Shagrir et Pitowski est constitué de deux ordinateurs modernes T_A et T_B pouvant communiquer entre eux. Grâce aux propriétés des espaces-temps de Malament-Hogarth, au moment où T_A aura effectué un nombre fini d'étapes calculatoires, T_B aura, quant à lui, calculé un nombre infini d'étapes. Maintenant, voici la procédure permettant à HM de calculer la fonction arrêt h des machines de Turing :

1. Nous commençons par fournir la donnée en entrée n à T_A qui la transmet à T_B. T_B est une machine de Turing universelle, c'est-à-dire que son programme lui permet de simuler le calcul de la n-ième machine de Turing opérant sur n.
2. Lors du calcul, si T_B s'arrête à un moment donné, ce dernier envoie immédiatement un signal à T_A. Si T_B ne s'arrête jamais, il n'envoie jamais de signal à T_A.
3. Enfin, lorsque que T_B aura effectué un nombre infini d'étapes en un temps fini, T_A inscrira 1 s'il a reçu un signal de T_B et insicrira 0 autrement. Par conséquent, HM a la capacité de calculer la fonction non Turing-calculable h définie par $h(n) = 1$ si la n-ième machine de Turing s'arrête sur n et $h(n) = 0$ autrement.

Les tentatives de construction d'une O-machine sont par ailleurs fondées sur la stratégie E, c'est-à-dire sur l'information non calculable issue de la nature. Dans ce but, il est tout d'abord nécessaire de localiser cette information dans la nature. L'idée de trouver cette information à partir de l'aléatoire quantique provient à la fois du modèle standard de la physique quantique (l'interprétation de Copenhague) et des travaux de Richard Feynman. D'une part, le modèle standard postule l'aléatoire quantique via le postulat de Born [8]. D'autre part, Feynman conclut dans son article, *Simulating Physics with Computers,* qu'« il est impossible de représenter les résultats de la mécanique quantique avec un dispositif classique universel [une machine de Turing] » [Feynman 1982, 476]. Plus récemment, Calude a proposé de concevoir un hyper-modèle physique qui pourrait utiliser l'aléatoire quantique comme un oracle et ainsi dépasser les limites calculatoires de la machine de Turing [Calude 2004].

7. Dans ce qui suit, la fonction arrêt h est définie de façon différente de celle définie dans la section 1. Soient $M_1, M_2, \ldots, M_m$ une énumération des machines de Turing, la fonction h est maintenant définie par $h(n) = 1$ si la n-ième machine de Turing s'arrête sur n et $h(n) = 0$ autrement.

8. Voici une formulation possible de ce postulat : lorsqu'un système quantique clos qui est dans l'état $V = (v_{1,1}, v_{2,1}, \ldots, v_{n,1}^T)$ est mesuré, la mesure conduit au résultat i avec une probabilité égale à $|v_{i,1}|^2$.

La stratégie de Calude consiste à fixer sur un ordinateur un appareil capable de générer une suite de nombres aléatoires via un processus quantique. Un tel appareil a été conçu par l'entreprise « id Quantique » et se nomme « Quantis » [9]. Quantis exploite un processus quantique élémentaire qui permet de générer une suite de nombres aléatoires [Jennewein, Achleitner, Weihs *et al.* 2000]. La lumière est en effet composée de particules élémentaires appelées « photons » qui, dans certaines situations, affichent un comportement aléatoire. La projection de ces derniers sur un miroir semi-transparent en est un exemple. Plus précisément, un photon généré par une source qui est projetée vers un mirroir semi-transparent a 50 % de chance d'être réfléchi par le mirroir et 50 % de chance de le traverser. En associant à ces deux évènements (réflexion, transmission) les *bits* 0 et 1, il est possible de traduire les mesures des positions des photons en une suite aléatoire de 0 et de 1. Un ordinateur équipé de Quantis pourrait en théorie produire une suite arbitrairement longue de *bits* quantiques aléatoires qui ne pourrait pas être générée par une machine de Turing [10]. Quantis serait donc vu comme un oracle qui fournit de l'information non calculable issue de la nature (via un processus quantique) dans le but de surpasser les capacités de la machine de Turing.

Les tentatives proposées afin de démontrer la thèse de l'hyper-calcul laissent supposer que le débat ne concerne que les physiciens. Cela n'est pas tout à fait exact, car la possibilité de construire un hyper-modèle physique soulève aussi des questions philosophiques qui n'ont pas encore trouvé de réponses unanimes. Je souhaite en particulier répondre dans le présent article à deux de ces questions.

La première question est liée aux enjeux de la thèse de l'hyper-calcul. Il est en effet important de se demander pourquoi les défenseurs de l'hyper-calcul considérent l'hyper-calcul comme un domaine ayant deux versants : l'un mathématique et l'autre physique. Cette question est justifiée par le fait que la construction physique d'un modèle de calcul, qu'il soit équivalent ou non à la machine de Turing, dépasse le cadre initial de la théorie du calcul effectif. D'une part, la thèse de Church-Turing n'énonce rien en ce qui concerne la puissance de calcul de la machine de Turing si elle est physiquement réalisée. D'autre part, même si l'hyper-calcul dépasse lui aussi le cadre du calcul effectif, ce dépassement reste à l'intérieur du versant mathématique contrairement à la possibilité de l'hyper-calcul dans le domaine physique. Par conséquent pourquoi sortir du cadre mathématique de l'hyper-calcul pour se tourner vers la construction physique d'un hyper-modèle ?

9. `http://www.idquantique.com/`.

10. Cette suite ne pourrait pas être générée par une machine de Turing uniquement dans le cas où elle serait infinie, car toute suite finie peut être générée par une machine de Turing.

3 Pourquoi est-il nécessaire de construire un hyper-modèle ?

La volonté de construire un hyper-modèle physique laisse supposer que cette construction apporte « quelque chose de plus » en ce qui concerne le calcul d'un hyper-modèle conceptuel. Plus précisément, bien que la suppression d'une certaine contrainte permette à un hyper-modèle de calculer une fonction non Turing-calculable, je pense que ce bénéfice calculatoire n'est pas gratuit. L'absence de cette contrainte fait en effet perdre à l'hyper-modèle une caractéristique fondamentale que possède la machine de Turing : la possibilité pour un humain d'avoir accès en principe aux résultats des calculs. Cependant, la construction physique d'un hyper-modèle permettrait de retrouver l'accès à ces résultats. Expliquons en détails ce raisonnement.

Tout d'abord, définissons ce que veut dire « avoir accès au résultat d'un calcul » :

Définition 3 (Avoir accès au résultat d'un calcul)

– Avoir accès au résultat d'un calcul, c'est pouvoir en principe avoir à disposition la donnée en sortie de ce calcul.

L'adjectif « en principe » signifie « indépendamment des contraintes de temps et d'espaces ». Autrement dit, on calcule en principe lorsque l'on dispose de toutes les ressources physiques nécessaires afin de mener à bien son calcul. Chaque fois que l'on manque d'une ressource (papier sur lequel écrire par exemple), il est possible de se réapprovisionner d'une nouvelle feuille. Néanmoins, même si ces ressources sont potentiellement infinies, le calcul en utilise une quantité finie.

À présent, montrons d'une part que l'on peut avoir accès aux résultats calculés par une machine de Turing et d'autre part qu'il n'est pas possible d'avoir accès aux résultats calculés par un hyper-modèle conceptuel.

En premier lieu, il est aisé de constater que l'on a en principe accès aux résultats calculés par une machine de Turing, car d'après la contrainte inscrite dans la définition d'une procédure effective « un être humain doit pouvoir suivre l'algorithme étape par étape, de la donnée initiale au résultat indépendamment des contraintes de temps et d'espaces mémoire » [Copeland 2002b, 1]. Ainsi, puisque les résultats d'une machine de Turing sont fournis après un nombre fini d'étapes, le calculateur peut avoir accès en principe à ces résultats.

En second lieu, expliquons pourquoi il est actuellement impossible d'avoir accès aux résultats d'un hyper-modèle conceptuel. Avoir accès aux résultats d'un hyper-modèle revient à satisfaire une de ces deux conditions :

1. Pouvoir suivre en principe les étapes de calcul de l'hyper-modèle de la donnée initiale au résultat.

2. Pouvoir accéder au même résultat que l'hyper-modèle sans suivre les étapes de son calcul.

Si l'hyper-modèle est la MTA ou l'O-machine, le calculateur humain doit avoir recours à l'infini afin de satisfaire la première condition. Il est en effet nécessaire de pouvoir exécuter un nombre infini d'étapes calculatoires en un temps fini pour suivre les étapes de la MTA ou bien mémoriser un nombre constitué d'une infinité de décimales si l'on souhaite suivre les étapes de l'O-machine. Il existe toutefois un argument convaincant allant à l'encontre de la thèse selon laquelle nous sommes capables de satisfaire la première condition. Cet argument consite à dire que le cerveau où sont effectués les calculs est une entité finie à la fois en espace (il existe un nombre fini de neurones et de liaisons synaptiques) et en temps (la vie d'un être humain est limitée). Cet argument, suffisant pour montrer que nous sommes incapables de satisfaire la première condition, ne l'est plus en ce qui concerne la seconde condition.

Il est tout à fait concevable que nous puissions avoir accès aux résultats d'une fonction non Turing-calculable sans pour autant avoir besoin de suivre chaque étape du calcul d'un hyper-modèle. Dans ce cas, le cerveau humain serait lui-même considéré comme un hyper-modèle physiquement réalisé pouvant avoir accès aux résultats d'une fonction non Turing-calculable. Cependant aucune tentative n'a à ce jour permis de prouver que le cerveau humain est capable d'hyper-calcul. Par exemple, Hava Siegelmann a proposé en 1995 un modèle mathématique du cerveau sous forme de réseaux de neurones artificiels capables selon elle de « calculer au-delà de la limite de Turing » [Siegelmann 1995]. Cet hyper-modèle fut néanmoins critiqué par Davis, car ces réseaux de neurones ne peuvent aller « au-delà de la limite de Turing » que si l'on suppose au préalable des nombres non Turing-calculables à l'intérieur de ces réseaux [Davis 2006]. Le modèle de Sielgelmann ne prouve donc pas que le cerveau est supérieur à la machine de Turing puisque la question de l'origine de ces nombres non Turing-calculables n'est pas expliquée [11].

À partir des deux arguments énoncés ci-dessus, celui concernant la finitude des ressources du cerveau et celui concernant les critiques du modèle de Siegelmann, je conclus qu'il est actuellement impossible qu'un calculateur humain puisse avoir accès aux résultats d'un hyper-modèle conceptuel. Toutefois, une solution possible consisterait à posséder un hyper-modèle physiquement construit. Il suffirait dans ce cas de fournir l'argument x à l'hyper-modèle qui a été construit et d'attendre un intervalle de temps fini pour qu'il fournisse le résultat $f(x)$, où f est une fonction non Turing-calculable. Nous pourrions alors

11. Le débat qui consiste à déterminer si le cerveau humain est supérieur à la machine de Turing n'est pas réduit à l'hyper-calcul. Ce débat a pour origine l'article de Turing intitulé « Computing Machinery and Intelligence » [Turing 1950] et concerne des domaines plus généraux, tels que l'intelligence artificielle et la philosophie de l'esprit.

avoir accès aux résultats d'un hyper-modèle sans pour autant avoir besoin de suivre chaque étape de son calcul.

Ce résultat, caractérisé par le lien entre la puissance de calcul d'un hyper-modèle et sa construction physique, a une importante conséquence pour la notion d'hyper-calcul, car il montre que certaines caractéristiques des hyper-modèles ne dépendent pas uniquement des domaines des mathématiques ou de la logique. Autrement dit, même si la puissance de calcul d'un hyper-modèle provient de l'absence de certaines contraintes formelles, la possibilité d'avoir accès aux résultats de fonctions non Turing-calculables se fonde sur des contraintes physiques. C'est donc aux sciences physiques et non aux mathématiques de décider s'il est possible d'avoir accès aux résultats d'une fonction non Turing-calculable.

Bien que la construction physique d'un hyper-modèle soit suffisante pour avoir accès aux résultats d'une fonction non Turing-caclulable, une telle construction n'est peut-être pas suffisante pour démontrer la thèse de l'hyper-calcul. Un problème épistémologique en particulier que j'appelerai « problème de la vérification » est en effet souvent invoqué. Ce dernier est énoncé dans le cas où nous disposerions d'un hyper-modèle physiquement construit et a pour conséquence l'impossibilité de vérifier que cet hyper-modèle calcule une fonction non Turing-calculable. En d'autres termes, même si nous disposions d'un hyper-modèle physiquement construit nous ne pourrions pas démontrer la thèse de l'hyper-calcul, car il serait impossible de vérifier que cet hyper-modèle est supérieur à la machine de Turing. Je propose dans la section suivante une analyse de ce problème dans le but de montrer qu'il ne remet pas en cause de façon explicite la thèse de l'hyper-calcul.

4 Le problème de la vérification

Le problème de la vérification a été souvent traité dans la littérature [Copeland 2002c], [Shagrir & Pitowski 2003], [Cleland 2004], [Davis 2006]. Ce problème n'a cependant jamais fait l'objet d'une analyse complète en ce qui concerne les problèmes qu'il soulève et les solutions que l'on peut y apporter. Je souhaite proposer une telle analyse dans cette section.

Le problème de la vérification fut énoncé par Copeland sous la forme d'une expérience de pensée :

> Il y a un problème épistémologique lié à l'hyper-calcul. Supposons que le génie de Laplace dise « Voici une boîte noire capable de résoudre le problème de l'arrêt » (le problème est soulevé quelle que soit la fonction non calculable par machine de Turing qui est considérée). Tapez n'importe quel entier x et la boîte fournira la valeur de la fonction arrêt $H(x)$ correspondante. Puisqu'il n'y

> a pas de méthode systématique pour calculer les valeurs de la fonction arrêt, vous n'avez pas de moyen de vérifier si la machine produit les réponses correctes ou non. Même simuler la machine de Turing en question ne vous aidera pas en général, parce qu'importe la durée durant laquelle vous regarderez la simulation, vous ne pourrez pas inférer que la machine ne s'arrêtera pas à partir du fait qu'elle ne s'est pas encore arrêtée. [Copeland 2002c, 490–491]

Copeland nous dit que même si nous disposons d'une machine physiquement construite capable d'hyper-calcul, il serait impossible de déterminer si cette machine est réellement un hyper-modèle. La raison en est que nous devons vérifier les résultats de la machine pour affirmer qu'elle est capable d'hyper-calcul. Or une telle vérification est impossible, car l'on ne détient pas de méthode systématique (procédure effective) nous permettant, à partir d'une donnée en entrée, de suivre chacune des étapes et d'arriver en un temps fini au résultat du calcul. Ainsi, toute tentative de construction d'un hyper-modèle est vouée à l'échec, car il n'y aura pas de moyen de prouver que l'on a réussi.

La vérification est d'autant plus un obstacle à la thèse de l'hyper-calcul, car elle est en partie possible dans le cas du calcul effectif. En effet, pour vérifier qu'une machine calcule une fonction, il est nécessaire de satisfaire les deux conditions suivantes :

1. Pouvoir vérifier que la machine fournit en sortie un résultat correct à partir d'une donnée en entrée.
2. Pouvoir vérifier que la machine calcule une fonction donnée, c'est-à-dire pouvoir vérifier, pour toute donnée en entrée, que la machine fournit en sortie un résultat correct.

D'une part, la première condition est satisfaite dans le cas du calcul effectif, mais n'est pas satisfaite dans le cas de l'hyper-calcul. Ce résultat découle directement de la section précédente : puisqu'un ordinateur moderne peut être étudié à partir de son modèle théorique qui est la machine de Turing, il est possible d'avoir accès aux résultats calculés par ce dernier en suivant le calcul étape par étape. Nous pouvons ainsi vérifier en principe les résultats calculés par un ordinateur. En revanche, nous ne pouvons pas procéder de la même manière avec un hyper-modèle, car nous ne sommes pas capables de suivre chaque étape de calcul afin de vérifier que le résultat final est correct.

D'autre part, la seconde condition n'est satisfaite ni par le calcul effectif ni par l'hyper-calcul, car il est impossible, uniquement à partir du comportement entrée-sortie d'une machine, de déterminer la fonction qui est calculée par la machine [Gold 1965]. Intuitivement, la raison en est que nous n'avons à notre disposition qu'un nombre fini de résultats, résultats qui pourront toujours s'accorder avec d'autres fonctions. Par exemple, si l'on fait l'hypothèse que la fonction calculée par la machine est $f(x) = 2x$ et que les 3^{100} premiers résultats s'accordent avec f, il sera toujours possible que la prochaine donnée

en entrée soit multipliée par 3. La fonction calculée par la machine sera donc différente de f.

Il est toutefois possible dans le cas du calcul effectif de falsifier certaines hypothèses contradictoires avec nos observations afin de se rapprocher de l'identification de la fonction calculée par l'ordinateur. Par exemple, les résultats $f(1) = 2, f(2) = 4, f(3) = 9$ falsifient notre hypothèse que la fonction calculée est $f(x) = 2x$. Mais dans le cas où la machine est un hyper-modèle, nous sommes incapables de falsifier l'hypothèse selon laquelle la fonction calculée est non Turing-calculable, car si nous faisons une telle hypothèse, notre nombre fini d'observations pourra toujours s'accorder avec une fonction Turing-calculable. Dans le cas contraire, nous serions parvenu à identifier les résultats d'une fonction non Turing-calculable et donc à la calculer.

Cette impossibilité de déterminer si une machine est capable d'hyper-calcul fut utilisée par Davis dans le but de montrer que la construction d'un hyper-modèle est un mythe [Davis 2006]. Les arguments énoncés ci-dessus auraient tendances à donner raison à Davis s'il n'existait pas d'autres réponses possibles au problème de la vérification.

Certains auteurs ont en effet proposé une réponse à ce problème dans le but de défendre la thèse de l'hyper-calcul [Cleland 2004, 223], [Shagrir & Pitowski 2003, 99]. Voici leur raisonnement :

1. Dans la pratique, la vérification des ordinateurs modernes est impossible tout comme la vérification des hyper-modèles.
2. Pourtant nous considérons en général que les ordinateurs calculent telle ou telle fonction.
3. Par conséquent, le calcul ne présuppose pas la vérification et la vérification des résultats calculés par un hyper-modèle ne doit pas être une condition nécessaire au calcul de fonctions non Turing-calculables.

Expliquons maintenant en détails ce raisonnement. Il consiste à dire que d'un point de vue pratique, la vérification n'est pas une condition nécessaire au calcul d'une fonction. Les auteurs différencient pour cela la vérification en principe de la vérification en pratique :

1. La vérification en principe fait abstraction des ressources calculatoires.
2. La vérification en pratique prend en compte les ressources calculatoires.

Leur principal argument est que nous considérons que les ordinateurs actuels calculent des fonctions, même s'il est impossible de vérifier en pratique les résultats calculés par ces ordinateurs.

Dans un premier temps, pourquoi est-il impossible en pratique de vérifier les résultats calculés par ordinateurs ? La réponse en est qu'aucune des deux conditions à satisfaire pour vérifier qu'un ordinateur calcule une fonction n'est satisfaite en pratique (conditions énoncées ci-dessus).

Montrons ce résultat à l'aide d'un exemple précis. Soit p la fonction définie par $p(n)$ = la n-ième décimale de l'expansion de π. Supposons par ailleurs que

nous disposons d'un ordinateur capable de calculer cette fonction, c'est-à-dire qui est capable de calculer la n-ième décimale de l'expansion de π. Pouvons-nous vérifier en pratique que cet ordinateur calcule p ?

Premièrement, pouvons-nous vérifier la fonction pour un de ses arguments ? Pour des valeurs de n comprises entre 1 et 20, il est encore possible avec de la patience de vérifier « à la main » les résultats calculés par l'ordinateur. Cependant, dans le cas où l'on souhaiterait vérifier que la décimale de π récemment calculée par ordinateur pour $n = 10^{12}$ est bien égale à 5, la vérification manuelle se trouve inefficace à cause d'un manque de temps. Par conséquent, il nous est impossible en pratique de vérifier les résultats calculés par un ordinateur pour certains arguments d'une fonction, résultats qui demandent trop de ressources calculatoires pour être vérifiés.

Deuxièmement, puisqu'il existe une infinité de résultats $p(n)$ avec $n \in \mathbb{N}$, nous pouvons vérifier qu'un nombre fini de résultats qui pourront toujours s'accorder avec une autre fonction [Gold 1965]. Il est donc impossible de vérifier en pratique qu'un ordinateur calcule une fonction donnée.

Pourtant, même si on est en pratique incapable de vérifier que notre ordinateur de bureau calcule correctement une fonction telle que l'addition de deux entiers, on serait tout de même tenté de dire que l'ordinateur calcule quand même cette fonction. La preuve en est que notre univers technique se présente aujourd'hui comme un univers d'ordinateurs qui sont indispensables au fonctionnement de notre société. Ils sont le fondement de notre économie (système bancaire), de nos moyens de transport aériens, ainsi que de nos moyens de production d'énergies (électrique et nucléaire). Nous accordons par conséquent une véritable confiance en ce qui concerne la fiabilité de nos ordinateurs.

Le raisonnement des défenseurs de l'hyper-calcul consiste à dire d'une part que, puisque nous avons confiance dans le calcul de nos ordinateurs, la vérification n'est pas une condition nécessaire au calcul de fonctions Turing-calculables. La vérification n'est pas nécessaire, car nous pouvons à l'aide de méthodes empiriques et théoriques (relations causales probabilistes, suppositions contrefactuelles fondées sur les lois physiques) affirmer de manière plausible qu'un ordinateur calcule une fonction Turing-calculable, même si une parfaite vérification est impossible. Leur raisonnement consiste à dire d'autre part que de tels tests pourraient exister dans le cas de l'hyper-calcul, tests grâce auxquels nous pourrions affirmer qu'un hyper-modèle physiquement construit calcule une fonction non Turing-calculable.

De ce point de vue, le problème de la vérification n'est pas une expérience de pensée comme dans la définition de Copeland, mais bien une « hypothèse empirique » [Cleland 2004, 224]. Autrement dit, le problème ne pourra pas être résolu tant que l'on supposera qu'un hyper-modèle est construit ; nous devons au contraire disposer réellement de l'hyper-modèle, afin de pouvoir faire des tests empiriques et ainsi affirmer de façon plausible qu'il calcule une fonction non Turing-calculable.

Pour terminer, clarifions ce dernier point en prenant l'exemple de l'O-machine physique proposée par Calude. Rappelons que cet hyper-modèle utilise l'aléatoire quantique pour produire une suite de nombres aléatoires qui ne peut pas être générée par une machine de Turing.

L'obstacle principal pour déterminer si cet hyper-modèle calcule une fonction non Turing-calculable repose sur le fait qu'il n'est pas encore prouvé que la physique quantique soit aléatoire puisque cette physique repose sur le postulat de Born. Plus précisément, il existe deux types distincts de processus aléatoires : les processus aléatoires et les processus pseudo-aléatoires. Les processus pseudo-aléatoires génèrent des suites de nombres à partir de méthodes pseudo-aléatoires (par exemple la méthode des congruences linéaires), nombres qui « paraissent » aléatoires, mais qui sont en réalité obtenus à l'aide d'algorithmes. Ainsi, dans le cas où les processus quantiques seraient des processus pseudo-aléatoires, une machine de Turing serait capable de les simuler, car les machines de Turing utilisant des algorithmes pseudo-aléatoires sont équivalentes aux machines de Turing standards [De Leeuw, Moore, Shannon *et al.* 1956, 183–212].

Résoudre le problème de la vérification de manière empirique consisterait à augmenter la plausibilité d'affirmer que l'O-machine calcule une fonction non Turing-calculable. Cela consisterait à faire des tests sur les suites aléatoires dans le but de montrer avec une forte probabilité que ces dernières ne sont pas pseudo-aléatoires. Si de tels tests peuvent être faits et que nous arrivons à la conclusion qu'il est probable que la suite est aléatoire, alors nous pouvons affirmer avec une forte plausibilité que l'hyper-modèle génère une suite dont les éléments sont les résultats d'une fonction non Turing-calculable. L'inconvénient est qu'actuellement les méthodes les plus avancées de génération de nombres pseudo-aléatoires produisent des suites qui, même quand elles sont soumises à toute sorte de tests, sont pratiquement impossibles à distinguer d'une suite de nombres aléatoires. Cependant, nous ne pouvons pas dénier la possiblité de posséder un jour des raisons sérieuses de croire qu'une suite est aléatoire. De façon équivalente, nous ne pouvons pas dénier la possibilité de posséder un jour des raisons sérieuses de croire qu'une machine physiquement construite est capable d'hyper-calcul.

Conclusion

Dans cet article, j'ai tenté dans un premier temps de montrer quels étaient les enjeux liés à la thèse de l'hyper-calcul. Présenter ces enjeux est essentiel puisque l'hyper-calcul, qui a pour origine le domaine de la logique, est par définition, étranger aux considérations d'ordre physique. J'ai défendu que les raisons de construire un hyper-modèle étaient plus fondamentales que les raisons de construire un ordinateur. Contrairement aux résultats calculés par

ordinateurs pour lesquels un accès en principe est possible à partir du modèle de la machine de Turing, il est impossible d'accéder en principe aux résultats de fonctions non Turing-calculables via un hyper-modèle conceptuel. Par conséquent, un hyper-modèle physiquement construit est plus qu'un outil pouvant améliorer nos performances dans le but d'accéder aux résultats (temps de calcul par exemple), c'est le moyen indispensable pour avoir accès à ces résultats. L'hyper-calcul doit ainsi être considéré comme un sujet transdisciplinaire qui doit être étudié par des logiciens et des physiciens, mais aussi par des philosophes ayant pour fonction de dégager ses enjeux qui vont au-delà du développement d'une science particulière.

La nécessité d'une analyse philosophique est d'autant plus importante puisqu'elle permet d'apporter des réponses à certains problèmes énoncés contre la thèse de l'hyper-calcul. En particulier, j'ai traité dans un second temps du problème épistémologique de la vérification dans le but d'en apporter des réponses. J'ai tout d'abord expliqué que ce problème fut énoncé à l'origine comme une expérience de pensée qui considère un type particulier de vérification, à savoir une vérification en principe. Ce problème a pour conséquence l'impossibilité de vérifier qu'une machine est un hyper-modèle dans le cas où l'on souhaiterait démontrer la thèse de l'hyper-calcul. J'ai ensuite essayé de montrer qu'une réponse possible était de considérer une vérification en pratique. On affirme en effet généralement que les ordinateurs calculent des fonctions bien qu'une telle vérification soit impossible. Cette affirmation a pour origine une confiance fondée sur des tests empiriques qui permettent de vérifier avec une certaine plausibilité qu'un ordinateur calcule une fonction. De la même manière, il pourrait s'avérer possible dans le cas des hyper-modèles que nous disposions un jour de techniques permettant d'affirmer avec une forte probabilité qu'une fonction non Turing-calculable a été calculée. Par conséquent, le problème de la vérification ne doit plus être énoncé comme une expérience de pensée, mais doit être considéré comme un problème empirique.

Remerciements

Je souhaite tout d'abord remercier à la fois les éditeurs et les relecteurs pour leurs commentaires qui m'ont été d'une grande aide. Je remercie également Anouk Barberousse pour ses remarques toujours éclairantes ainsi que pour avoir soutenu ce travail à l'aide du projet de recherche Compuphys (Agence Nationale de la Recherche (ANR-08-JCJC-0035-01)).

Bibliographie

Benacerraf, Paul

1962 Super-Tasks and the Modern Eleatics, *Journal of Philosophy*, 59, 765–785.

Calude, Cristian S.

2004 Algorithmic randomness, quantum physics, and incompleteness, Rap. tech. 248, Department of Computer Science, The University of Auckland, New Zealand.

Church, Alonzo

1936 An unsolvable problem of elementary number theory, dans *The Undecidable (1965)*, édité par Davis, M., Mineola, New York : Dover, 89–107.

Cleland, Carol

2004 The concept of computability, *Theoretical Computer Science*, 317, 209–225.

Copeland, Jack

2002a Accelerating Turing Machine, *Minds and Machines*, 12, 281–301.

2002b The Church-Turing thesis, *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu.

2002c Hypercomputation, *Minds and Machines*, 12, 461–502.

Copeland, Jack & Proudfoot, Diane

1999 Alan Turing's forgotten ideas in computer science, *Scientific American*, 208, 76–81.

Davis, Martin

2006 The Myth of Hypercomputation, dans *Alan Turing : Life and Legacy of a Great Thinker*, édité par Teusche, C., Berlin : Springer, 195–212.

De Leeuw, K., Moore, E.F., Shannon, C.E. & Shapiro, N.

1956 *Computability by Probabilistic Machines*, Automata Studies : Princeton University Press.

Earman, John & Norton, John D.

1993 Forever is a Day : Supertasks in Pitowski and Malament-Horgarth spacetimes, *Philosophy of Science*, 60, 22–42.

Feynman, Richard P.

1982 Simulating physics with computers, *International Journal of Theoretical Physics*, 21(6–7), 467–488.

Gold, Mark

1965 Limiting recursion, *The Journal of Symbolic Logic*, 30, 28–48.

Hagar, Amit & Korolev, Alexandre

2006 Quantum hypercomputability ?, *Minds and Machines*, 16(1), 87–93.

Hogarth, Mark

1992 Does general relativity allow an observer to view an eternity in a finite time ?, *Foundations of Physics Letters*, 5, 173–181.

Jennewein, Thomas, Achleitner, Ulrich, Weihs, Gregor, Weinfurter, Harald & Zeilinger, Anton

2000 A fast and compact quantum random number generator, *Review of Scientific Instruments*, 71, 1675–1680.

Kieu, Tien D.

2002 Quantum hypercomputation, *Minds and Machines*, 12(4), 541–561.

Kleene, Stephen C.

1936 General recursive functions of natural numbers, dans *The Undecidable*, édité par Davis, M., Mineola : Dover, 236–253, 1965.

Loff, Bruno & Costa, José Félix

2009 Five views of hypercomputation, *International Journal of Unconventional Computing*, 5, 193–207.

Pitowski, Itamar

1990 The physical Church thesis and physical computational complexity, *Iyyun*, 39, 81–99.

Post, Emil

1941 Absolutely unsolvable problems and relatively undecidable propositions—account of an anticipation, dans *The Undecidable*, édité par Davis, M., Mineola, New York : Dover, 340–433, 1965.

Rogers, Hartley

1987 *Theory of Recursive Functions and Effective Computability*, The MIT Press.

Shagrir, Oron & Pitowski, Itamar

2003 Physical hypercomputation and the Church-Turing thesis, *Minds and Machines*, 13, 87–101.

Shepherdson, John C. & Strurgis, Howard E.

1963 Computability of recursive functions, *Journal of the Association of Computing Machinery*, 10, 217–255.

Shoenfield, Joseph R.

1967 *Mathematical Logic*, Reading Mass. : Addison-Wesley.

Siegelmann, Hava T.

1995 Computation beyond the Turing limit, *Science*, 268, 545–548.

Stone, Harold S.

1972 *Introduction to Computer Organization and Data Structures*, New York : McGraw-Hill.

Thomson, James F.

1954 Tasks and super-tasks, *Analysis*, 15(1), 1–10.

Turing, Alan

1936 On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society*, 2(42), 230–265.

1939 Systems of logic based on the ordinals, dans *The Undecidable*, édité par Davis, M., Mineola, New York : Dover, 154–222, 1965.

1950 Computing machinery and intelligence, *Mind*, 59(36), 433–460.

Van Bendegem, Jean Paul

1994 Ross' paradox is an impossible super-task, *British Journal for the Philosophy of Science*, 45, 743–748.